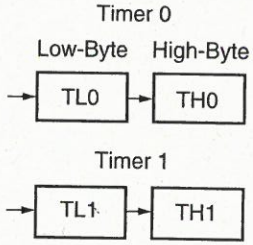
16 Die Zähler/Zeitgeber Timer 0 und 1

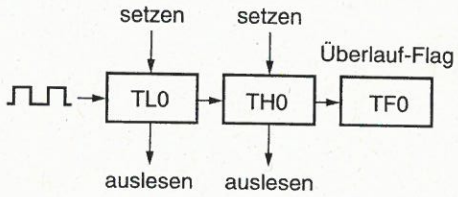
Timer 0 und Timer 1 bestehen aus jeweils zwei 8-Bit-Vorwärtszählern. Die Zähler lassen sich, je nach gewähltem Modus, als 8-Bit oder als 16-Bit-Zähler verwenden. Die 16-Bit-Zähler ergeben sich aus der Reihenschaltung zweier 8-Bit-Zähler.

Die 8-Bit-Zähler-Register sind die Spezial-Funktions-Register TL0 und TH0 für Timer 0 und TL1 und TH1 für Timer 1.



Die Zähler-Register lassen sich auf Anfangswerte setzen. Ab diesen Werten zählen sie nach dem Starten des Zählers aufwärts. Läuft der Zähler über, wird das Überlauf-Flag gesetzt. Nach einem Überlauf lässt sich der Zähler wieder auf einen Anfangswert setzen. Geschieht dies nicht, beginnt er bei dem Wert null.

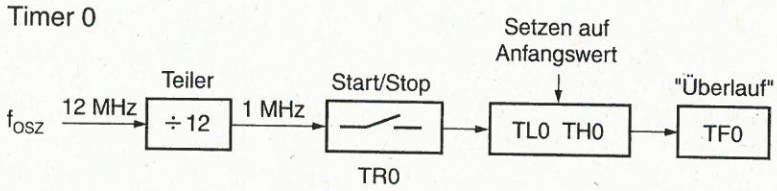
Timer 0 als 16-Bit-Zähler:



Das Überlauf-Flag kann einen Interrupt auslösen oder vom Programm abgefragt werden. Bei einer Abfrage per Programm ist es auch durch das Programm wieder zu löschen.

Der jeweilige Zählerstand lässt sich auch aus den Registern auslesen.

16.1 Einsatz der Timer als Zeitgeber



Bei einer Verwendung der Timer als Zeitgeber werden sie vom Systemtakt hochgezählt. Die Periodendauer des Systemtaktes wird über den Zähler addiert. Je nach Anfangswert des Zählers ergibt sich bis zum Überlauf eine bestimmte Zeitverzögerung. Diese Zeitverzögerung ist nicht abhängig vom laufenden Programm. Der Timer läuft, getaktet vom Systemtakt, als eigenständige Hardware-Baugruppe. Die Zeitverzögerung hängt also nur von der Quarzfrequenz ab.

Läuft der Zähler über, setzt er sein Überlauf-Flag TF0 oder TF1 auf 1. Damit lässt sich ein Interrupt auslösen, mit dem das laufende Programm unterbrochen und ein Unterprogramm als Interrupt-Service-Routine gestartet werden kann.

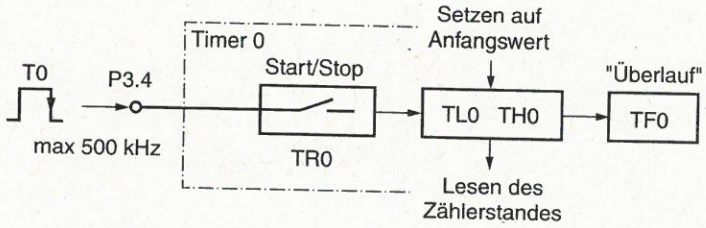
Wird nicht im Interruptbetrieb gearbeitet, wird das Überlauf-Flag bei jedem Programmzyklus abgefragt. Ist es gesetzt, wird darauf reagiert und das Flag zurückgesetzt.

Die Überlauf-Flags sind Bits in bestimmten Spezial-Funktions-Registern. Sie sind bitadressierbar.

Einstellen der Zeit und starten: z. B. Timer 0

1. Taktsignal stoppen:
   ```
   CLR TR0
   ```
2. Anfangswerte in beide Timerbytes laden:
   ```
   MOV TL0,#ANFL
   MOV TH0,#ANFH
   ```
3. Überlauf-Flag rücksetzen:
   ```
   CLR TF0
   ```
4. Taktsignal starten:
   ```
   SETB TR0
   ```

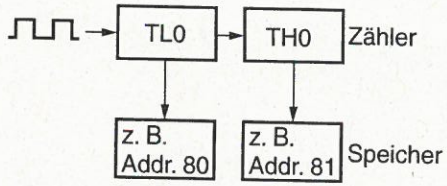
16.2 Einsatz der Timer als Ereigniszähler



Werden die Timer als Ereigniszähler eingesetzt, erhalten sie ihre Zählimpulse von einem externen Geber. Es werden die negativen Flanken der Eingangsimpulse gezählt. Dabei arbeiten die Zähler unabhängig vom laufenden Programm.

Lesen des Zählerstandes

Der 16-Bit-Zähler wird aus zwei Registern gebildet, die nacheinander auszulesen sind. Das ist kein Problem, wenn der Zähler während des Auslesens über TR0 angehalten werden kann.



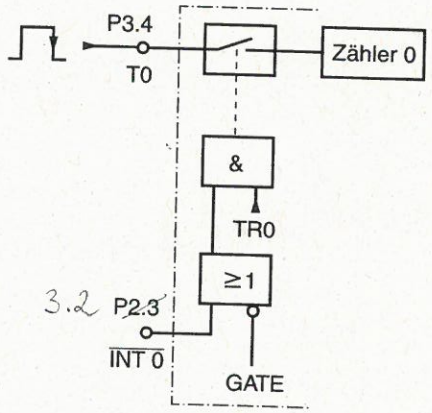
Will man bei laufendem Zähler den Zählerstand auslesen, ist zu berücksichtigen, dass während des Auslesens Zählimpulse eintreffen können, die den Zählerstand verändern. Es ist daher zu kontrollieren, ob während des Auslesens ein Übertrag in das höherwertige Byte aufgetreten ist. Ist das der Fall, ist das Lesen zu wiederholen.

Programm zum Auslesen des Zählerstandes des Timers 0:

```
LES_Z16:
    MOV 81,TH0; H-Byte nach 81
    MOV 80,TL0; L-Byte nach 80
    MOV A,TH0 ;TH_NEU = TH_ALT?
    CJNE A,81,LES_Z16 ;Nein:
    RET    ;Ja:
```

Freigabe des Zählers

Am Beispiel des Timers 0:
Der Zähler lässt sich über die Bits TR0 und GATE in Spezial-Funktions-Registern oder über ein externes Signal an Portpin P3.2 freigeben.



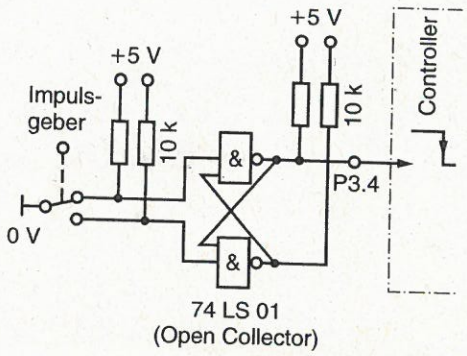
Interne Freigabe:
Externe Freigabe über P3.2 = 1:

```
GATE = 0; TR0 = 1
GATE = 1; TR0 = 1
```

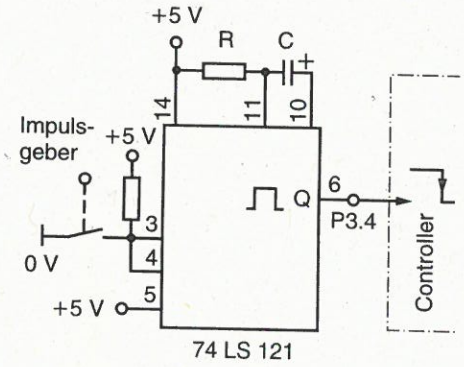
Entprellen des Zählimpulses

Kommt der Eingangsimpuls für den Zähler von einem mechanischen Kontakt, muss er entprellt werden. Das kann nicht per Software geschehen, sondern durch eine Hardware vor dem Controller-Eingang. Damit lassen sich auch kurze Störimpulse herausfiltern.

Entprellen mittels eines Flipflops:



Entprellen über eine monostabile Kippstufe:



16.3 Einstellen der Timer-Funktion

Die Timer-Funktion wird mithilfe des Timer-Modus-Registers TMOD eingestellt.

Timer-Modus-Register TMOD: Adresse 89H

TMOD nicht bitadressierbar — Resetwert 00h

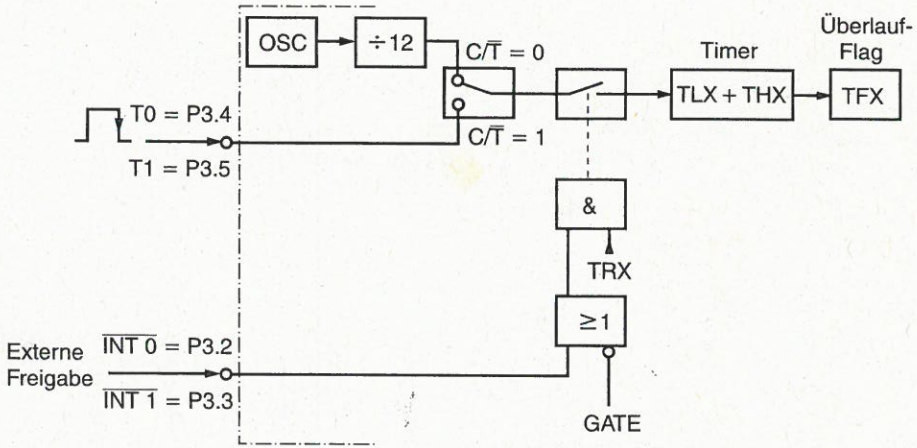
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
7							0
Timer 1				Timer 0			

Bedeutung der einzelnen Bits des Timer-Modus-Registers:

Bit		Funktion
GATE		Umschaltung interne/externe Freigabe:
0		Interne Freigabe über TR0 bzw. TR1
1		Externe Freigabe über P3.2 bzw. P3.3 plus interne Freigabe über TR0 bzw. TR1.
C/$\overline{T}$		Umschaltung Zeitgeber oder Zähler:
0		Funktion als Zeitgeber
1		Funktion als Zähler
M1	M0	Wahl des Arbeitsmodus: (Es werden nur die zwei gebräuchlichen Arbeitsmodi beschrieben)
0	1	Modus 1: TL0 und TH0 bzw. TL1 und TH1 bilden einen 16-Bit-Zähler
1	0	Modus 2: TL0 bzw. TL1 bilden einen 8-Bit Auto-Reload-Zähler. Bei Überlauf wird der im High-Byte TH0 bzw. TH1 stehende Wert in das Low-Byte kopiert. Das High-Byte bleibt unverändert.

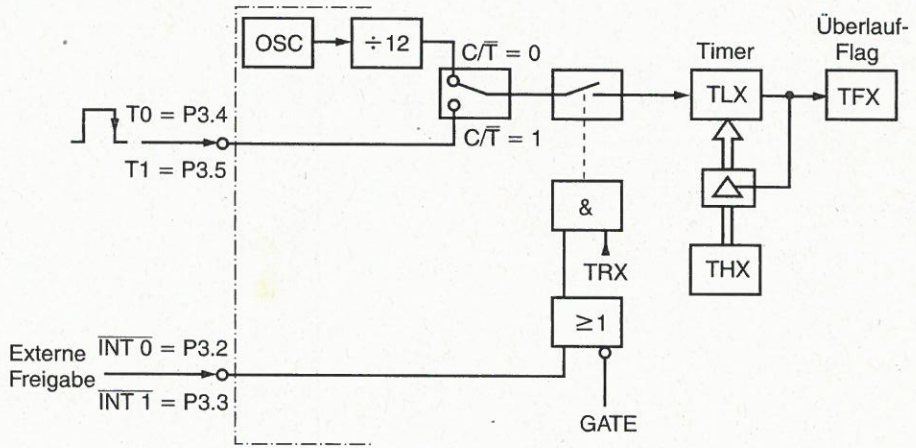
Blockschaltbild der Timer 0 und 1 in Modus 1

16-Bit-Timer/Counter:



Blockschaltbild der Timer 0 und 1 in Modus 2

8-Bit-Timer/Counter mit Auto-Reload:



16.4 Steuern der Timer

Die Timer werden über die Bits TF und TR im Timer-Control-Register TCON gesteuert.

Timer-Control-Register TCON: Adresse 88H

TCON bitadressierbar — Resetwert 00h

TF1	TR1	TF0	TR0				
7							0

Bedeutung der einzelnen Bits des Timer-Control-Registers:

Bit	Funktion
TF0 TF1	Timer-Überlauf-Flag: Das Flag wird beim Überlauf des Timers gesetzt. Beim Einsprung in die zugehörige Interrupt-Adresse wird es automatisch rückgesetzt. Bei Abfrage im Programmzyklus muss es durch das Programm gelöscht werden.
TR0 TR1 1 0	Timer-Run-Flag = Timer Freigabe: Das Flag muss durch das Programm gesetzt oder gelöscht werden. Timer startet Timer stopt

Beispiel 16.1

Initialisieren Sie den Timer 1 als Zeitgeber. Es soll eine Zeit für ein Blinklicht erzeugt werden. Timer 0 ist nicht zu beeinflussen.

Timer-Modus einstellen:

TMOD nicht bitadressierbar

GATE	C/T	M1	M0				
0	0	0	1	x	x	x	x

Assemblerprogramm:

```
INIT_T1:  ANL TMOD,#0Fh; 16-Bit
          ORL TMOD,#10h  Zähler
          RET
```

Der Timer soll vom laufenden Programm auf einen Zeitwert gesetzt und auf Überlauf abgefragt werden. Nach jedem Zeitabschnitt wird das Blinklicht an P5.0 umgeschaltet.

Assemblerprogramm:

```
BLINK:
    JNB TF1,BL1;Timer Überlauf?
    CLR TR1; Ja: Timer stoppen
    MOV TL1,#ZL1; Timer setzen
    MOV TH1,#ZH1
    CLR TF1
      ; Überlauf rücksetzen
    CPL P5.0
      ; Blinklicht umschalten
    SETB TR1; Timer starten
BL1: RET
```

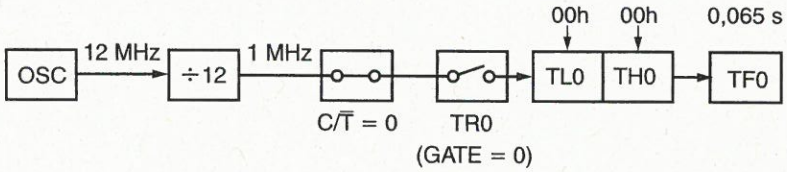
Übung 16.1

Initialisieren Sie Timer 0 als 16-Bit-Zähler. Es sollen externe Impulse gezählt werden. Die Eingangsimpulse vom Geber sind über eine monostabile Kippstufe entprellt. Über P3.2 soll sich der Zähler auf null stellen lassen. Zeigen Sie den Zählerstand auf Port 4 und 5 an (siehe „Lesen des Zählerstandes“).

16.5 Anwendung als Zeitgeber

Timer 0 soll als Zeitgeber für ein Lauflicht eingesetzt werden. Die Zeiten sollen einstellbar sein und in einem Bereich liegen, der mit den Augen erkennbar ist.

Die Basiszeit liefert der 16-Bit-Timer 0.



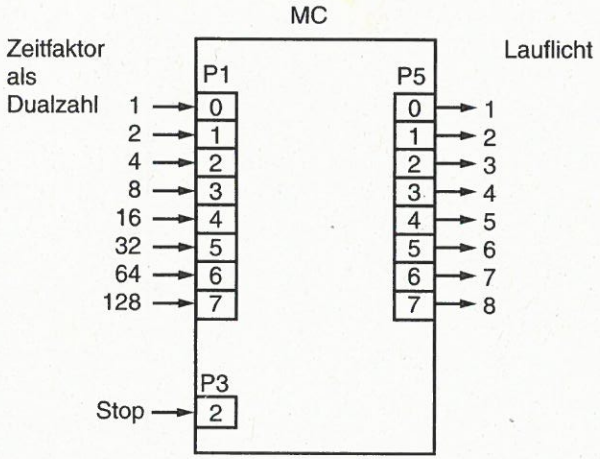
Bei Laden des Timers mit der Zahl 0000h ergibt sich bis zum Überlauf eine Zeit von 0,065 53 s, wenn der Controller mit einem 12-MHz-Quarz arbeitet.

Dieser Basiswert wird mit einem einstellbaren Zeitfaktor multipliziert.

$$\text{Zeit} = \text{Zeitfaktor} \times \text{Basiswert}$$

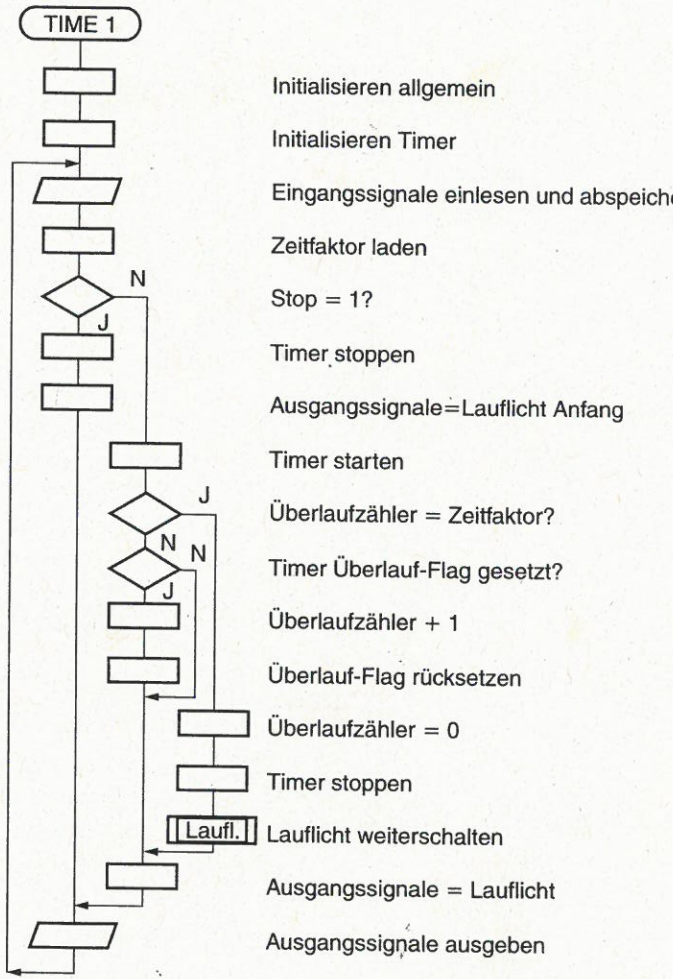
Dazu müssen die Timer-Überläufe gezählt werden. Stimmt der Zählerstand des Überlaufzählers mit dem eingestellten Zeitfaktor überein, ist die Zeit erreicht, und das Lauflicht schaltet um eine Stelle weiter.

Anschlussplan:



Ist das Stop-Signal = 1, soll die laufende Zeit unterbrochen werden und das Lauflicht in die Grundstellung gehen. Bei Stop = 0 setzt das Lauflicht die unterbrochene Zeit und Bewegung fort.

Programmablaufdiagramm:



Assemblerprogramm:

```
;********************** Timer als Zeitgeber ************************
;
;Timer 0 als Zeitgeber mit einstellbarer Zeit.
;Der Zeitfaktor wird auf Port 1 eingestellt.
;Nach abgelaufener Zeit soll ein Lauflicht an Port 5 um eine
;Stelle weiter schalten.
;Mit einem Stop-Signal an Port 3.2 ist das Lauflicht anzuhalten.
;
;Eingabe: Port 1
;         Bit 0 bis Bit 7: einstellbarer Zeitfaktor als Dualzahl
;         Port3
;         Bit 2: Stop
```

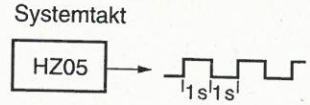
```
;                   Bei 1-Signal soll der Zeitgeber angehalten werden
;                   und das Lauflicht in die Ausgangsstellung gehen.
;                   Bei anschließendem 0-Signal soll das Lauflicht bei
;                   der unterbrochenen Stellung weiterarbeiten.
;Ausgabe: Port 5
;         Bit 0 bis Bit 7: Lauflicht
;
;*****************************************************************
controller 80535
Title "Timer als Zeitgeber, Datei TIME1.ASM"
org 8000h
;
;--------------------------- Zuordnungen ---------------------------
;
ein1 equ 20h                        ;Eingangssignalabbild Port 1
ein3 equ 21h                        ;Eingangssignalabbild Port 3
aus5 equ 22h                        ;Ausgangssignalabbild Port 5
zf equ 23h                          ;Zeitfaktor
uz equ 24h                          ;Überlaufzähler
lauf equ 25h                        ;Muster Lauflicht
stop equ ein3.2                     ;Schalter "Stop"
;
;--------------------- Initialisieren allgemein --------------------
;
mov p1,0ffh
mov p3,0ffh
mov uz,#00h
mov lauf,#01h                       ;Lauflicht Anfang
;
;---------------------- Initialisieren Timer0 ----------------------
;
anl tmod,#0f0h                      ;16 Bit Timer:
orl tmod,#01h                       ;Gate=0, C/T=0, M1=0, M0=1
mov tl0,#00h                        ;Timer nullstellen
mov th0,00h
;
;------------------------- Hauptprogramm -------------------------
;
ti5:    mov ein1,p1                 ;Eingangssignale einlesen
        mov ein3,p3
        mov zf,ein1                 ;Zeitfaktor nach ZF
        jnb stop,ti1                ;Stop = 1? Nein: nach ti1
        clr tr0                     ;Ja: Timer stoppen
        mov aus5,#01h               ;Ausgangssignale = Lauflicht Anfang
        ljmp ti2
ti1:    setb tr0                    ;Timer starten
        mov a,zf                    ;Überlaufzähler = Zeitfaktor?
        cjne a,uz,ti3               ;Nein: nach ti3
        mov uz,#00h                 ;Ja: Überlaufzähler nullstellen
        clr tr0                     ;Timer stoppen
        lcall laufli                ;Lauflicht weiterschalten
        ljmp ti4
ti3:    jnb tf0,ti2                 ;Timer Überlauf? Nein: nach ti2
        inc uz                      ;Ja: Überlaufzähler + 1
        clr tf0                     ;Überlauf-Flag nullstellen
ti4:    mov aus5,lauf               ;Ausgangssignale = Lauflicht
ti2:    mov p5,aus5                 ;Ausgangssignale ausgeben
        ljmp ti5                    ;zyklische Programmbearbeitung
;
;------------------- Unterprogramm Lauflicht ---------------------
;
laufli: mov a,lauf
        rl a                        ;Lauflicht 1 Stelle weiter
        mov lauf,a
        ret
;
;*****************************************************************
```

Beispiel 16.2

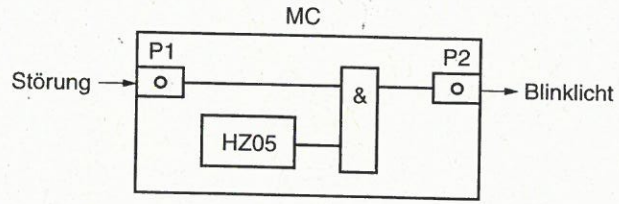
Systemtakt 0,5 Hertz

Bei vielen Automatisierungsgeräten wird dem Anwender ein Systemtakt mit einer bestimmten Frequenz zur Verfügung gestellt. Damit lassen sich Blinklichter, Warnhupen oder Zeitzähler ansteuern. Der Systemtakt wird von einem bestimmten Merkerbit ausgegeben, welches seinen Zustand mit einer vorgegebenen Frequenz ändert.

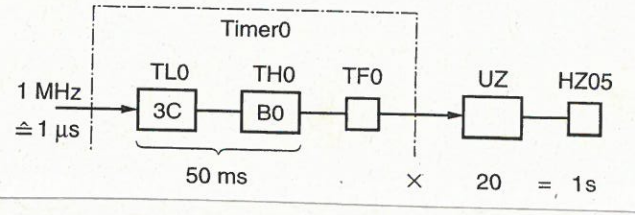
Hier soll ein Systemtakt von 0,5 Hz erzeugt werden. Das Merkerbit mit dem symbolischen Namen HZ05 muss dazu seinen Zustand jede Sekunde ändern.



Als Anwendung soll der Systemtakt HZ05 bei einer eintreffenden Störung am Port P1.0 ein Blinklicht an P5.0 einschalten.



Erzeugung des Systemtaktes HZ05



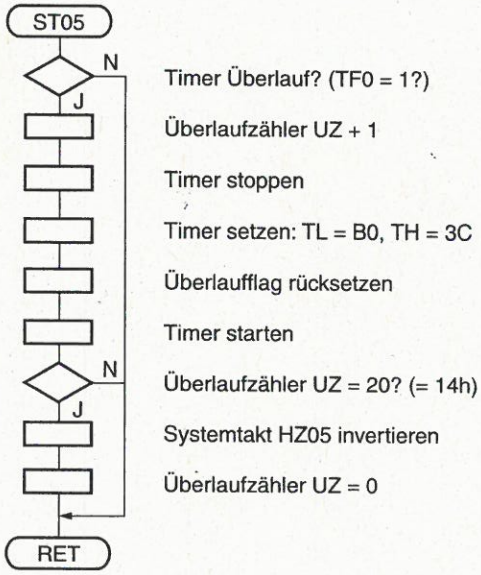
Um eine Zeit von 50 ms zu erhalten, muss der Timer auf die Dezimalzahl 15 536 dezimal oder 3C B0 hex gesetzt werden.

$$65\,536 - 50\,000 = 15\,536$$

Die 50 ms sind mit 20 zu multiplizieren, um auf eine Sekunde zu kommen. Dazu wird der Überlaufzähler mit dem symbolischen Namen UZ vom Überlaufflag TF0 von 0 bis 20 hochgezählt. Ist die 20 erreicht, wird das Bit HZ05 invertiert.

Unterprogramm SYSTA zur Erzeugung des Systemtaktes

Programmablaufdiagramm:



Assemblerprogramm:

```
;*****************************************************************
;Unterprogramm SYSTA.ASM zur Erzeugung eines Systemtaktes von 0,5 Hz
;an Bit HZ05.
;Das Unterprogramm benötigt   1 Merkerbit, Name HZ05
;                             1 Merkerbyte, Name UZ
;Initialisierung des Timers 0 als 16-Bit-Timer
;*****************************************************************
systa:jnb tf0,sy1
    inc uz
    clr tr0
    mov tl0,#0b0h
    mov th0,#3dh
    clr tf0
    setb tr0
    mov a,uz
    cjne a,#14h,sy1
    cpl hz05
    mov uz,#00h
sy1:ret
```

Übung 16.2

Schreiben Sie das Hauptprogramm für die in Beispiel 16.2 aufgeführte Anwendung des Systemtaktes HZ05 zur Ansteuerung eines Blinklichtes an Ausgabeport P5.0, wenn am Eingabeport P1.0 ein Signal „Störung“ anliegt.

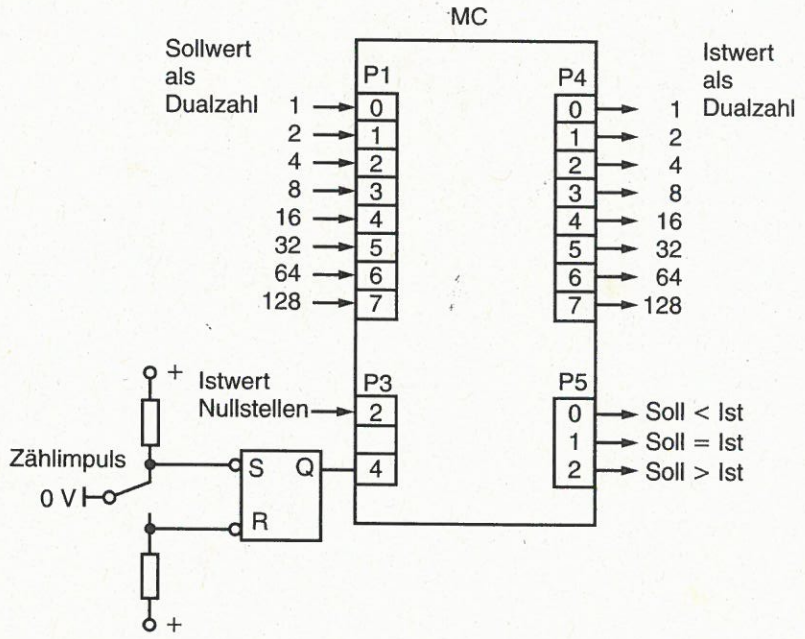
Denken Sie an die Zuweisungen und die Initialisierung für das benötigte Unterprogramm SYSTA.

16.6 Anwendung als Ereigniszähler

Bei einer digitalen Regelung wird ein eingestellter Sollwert mit einem Istwert verglichen. Der Istwert wird digital gebildet, indem eintreffende Zählimpulse von einem Geber einen Vorwärtszähler hochzählen.

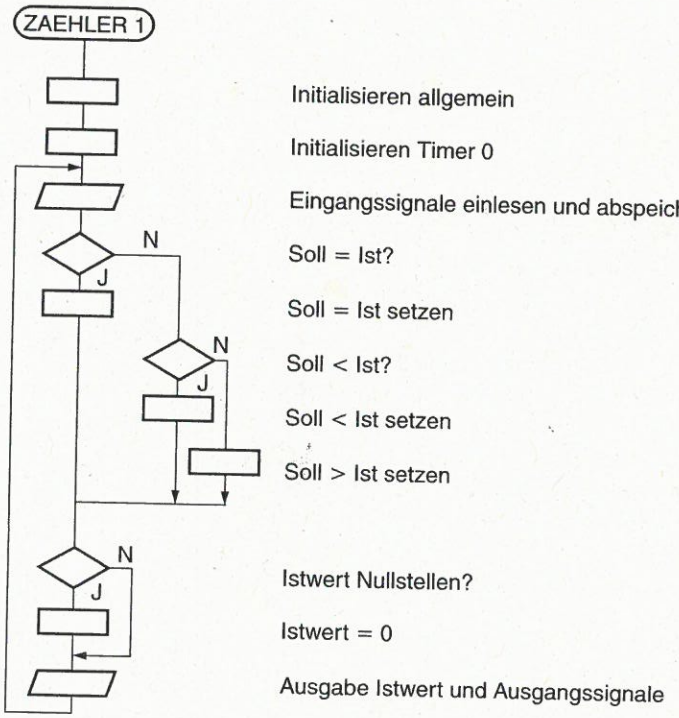
Die digitale Regelung könnte z. B. Teil eine Wegregelung sein, bei der pro Haltepunkt ein Zählimpuls erzeugt wird.

Anschlussplan:



Als Istwertzähler wird Timer 0 als 8-Bit-Reloadzähler eingesetzt. Der Reloadwert ist die Zahl 00h. Der Timer zählt die an P3.4 eintreffenden Impulse unabhängig vom laufenden Programm. Der Istwert des Timers wird vom laufenden Programm gelesen und bearbeitet.

Programmablaufdiagramm:



Assemblerprogramm:

```
;******************** Timer0 als Ereigniszähler *********************
;
Title "Ereigniszähler, Datei ZAEHLER1.ASM"
controller 80535
org 8000h
ljmp za_anf
;--------------------------- Zuordnungen ----------------------------
ein0 equ 20h                    ;Eingangssignalabbild 0
ein1 equ 21h                    ;Eingangssignalabbild 1
aus0 equ 22h                    ;Ausgangssignalabbild 0
;
;------------------- Initialisierung des Timer0 ---------------------
;
;Der Timer0 wird als 8-Bit-Reload-Counter eingesetzt.
;Der Reloadwert ist 00h.
;
init_t0: anl tmod,#0f6h         ;G=0, M0=0
         orl tmod,#06h          ;C/T=1, M1=1
         setb tr0               ;Zähler Ein
         setb p3.4              ;Vorbereitung p3.4 als Eingang
         mov th0,#00h           ;Reloadwert = 00h
         ret
```

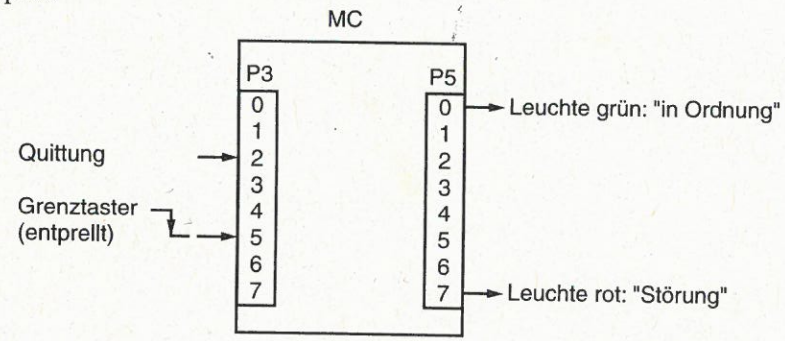
```
;------------------ Initialisierung der Steuerung -----------------
;
za_anf:  mov ein0,#00h          ;Eingangssignalabbild 0 = 0
         mov ein1,#00h          ;Eingangssignalabbild 1 = 0
         mov aus0,#00h          ;Ausgangssignalabbild 0 = 0
         mov p1,#0ffh           ;Eingabeports vorbereiten
         setb p3.2
         lcall init_t0          ;Timer0 initialisieren
;------------------------- Programm -------------------------------
za_5:    mov ein0,p1            ;Eingangssignale abspeichern
         mov ein1,p3
         mov a,ein0             ;Soll - Ist
         subb a,tl0
         jz za_1                ;Soll = Ist
         jc za_2                ;Soll < Ist
         setb aus0.2            ;Ausgabebit S>I auf 1
         ljmp za_3
za_1:    setb aus0.1            ;Ausgabebit S=I auf 1
         ljmp za_3
za_2:    setb aus0.0            ;Ausgabebit S<I auf 1
za_3:    mov c,ein1.2           ;Istwert 0-stellen?
         jnc za_4               ;Nein
         mov tl0,#00h           ;Istwert auf 00h stellen
za_4:    mov p4,tl0             ;Ausgabe Istwert auf Port 4
         mov p5,aus0            ;Ausgabe Steuersignale auf Port 5
         ljmp za_5              ;zyklische Programmbearbeitung
```

Übung 16.3

Schreiben Sie ein Assemblerprogramm für folgende Aufgabenstellung:

Ein Grenztaster liefert Eingangsimpulse. Wird er innerhalb von fünf Sekunden mehr als dreimal betätigt, liegt eine Störung vor. Im Normalbetrieb leuchtet eine grüne Lampe, im Störfall eine rote. Die Störung muss mit einer Taste quittiert werden.

Anschlussplan:



Als Zeitgeber wird Timer 0 verwendet. Die Basiszeit bis zum Überlauf beträgt 0,065 s. Die Überläufe sind zu zählen. Hat der Überlaufzähler die Zahl 77 erreicht, sind ca. 5 s vergangen.

Als Ereigniszähler wird Timer 1 eingesetzt. Hat der Ereigniszähler innerhalb 5 s mehr als drei Grenztaster-Betätigungen gezählt, liegt eine Störung vor, die angezeigt wird.

Programmablaufdiagramm: